Serdica J. Computing **5** (2011), 65–78

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

SPECIFICS IN APPLYING AGILE SOFTWARE METHODOLOGIES IN PORTAL SOLUTIONS

Nikolay Todorov, Avram Eskenazi

Abstract. Agile methodologies are becoming more popular in the software development process nowadays. The iterative development lifecycle, openness to frequent changes, tight cooperation with the client and among the software engineers are turning into more and more effective practices and respond to a higher extend to the current business needs. It is natural to raise the question which methodology is the most suitable for use when starting and managing a project. This depends on many factors—product characteristics, technologies used, client's and developer's experience, project type. A systematic analysis of the most common problems appearing when developing a particular type of projects—public portal solutions, is proposed. In the case at hand a very close interaction with various types of end users is observed. This is a prerequisite for permanent changes during the development and support cycles, which makes them ideal candidates for using an agile methodology. We will compare the ways in which each methodology addresses the specific problems arising and will finish with ranking them according to their relevance. This might help the project manager in choosing one or a combination of the methodologies.

ACM Computing Classification System (1998): D.2.9, K.6.3.
Key words: agile methodologies, XP, Scrum, ASD, FDD.

1. Introduction.

1.1. Agile methodologies. Agile methodologies are intended for use in developing software projects [1]. Several major methodologies exist—Extreme Programming (XP) [2], Scrum [3], Feature-Driven Development (FDD) [4], Adaptive Software Development (ASD) [5] and others. They try to reduce the risks by developing the projects in short periods of time called iterations, which usually last between one and four weeks. Each iteration is like a separate software project including all of the phases necessary to develop and deliver a new functionality—planning, analysis, design, coding, testing and documentation.

Agile methodologies prefer real time communication to written documentation. In most cases the whole team working on such a project is located at the same place. This includes the developers as well as their "customers" or "clients" (the people that define the product requirements).

Agile methodologies consider the creation of working software as a major success criterion. This goal combined with the preference for face-to-face communication generates as a result much less documentation compared to other methods, which sometimes is the reason to call them non-disciplined.

The common thing between the agile methodologies is that they always try to adapt the software process to the constantly changing conditions and requirements. All of them are a set of well-known practices combined in a way leading the project to a successful end. Not each of them covers the whole development lifecycle. Some concentrate only on particular phases. The same methodology is not always universally applicable in every domain where a software solution is needed. The same methodology is not even always appropriate in the same domain in two different cases. The successful application of a combination of different methodologies and their practices might result in building a successful process for a particular software project.

Each methodology is in a different phase of development. XP and Scrum are already completely built methodologies that are well documented and much material and feedback about them exists. Therefore we can define them as "Active". We define FDD and ASD as "In construction" as there is still scarce information about them and there is no detailed information or feedback about their application.

Although they are all based on similar concepts and principles, each of the agile methodologies looks upon the software development from a different angle. By using [6] we will make a comparison (Table 1) between them in the following aspects—**key points** that define the basic aspects and concept of the methodology, **characteristics** that differentiate them from the rest and **disadvantages** identified for them.

Table 1. Comparison of methodologies

Methodology	Key points	Characteristics	Disadvantages
XP	Oriented to the client development process, small teams, daily builds	Refactoring—constant improvement of the system to achieve better performance and adaption to changes	Although individual practices are suitable in most cases, the overall project management practices are incompletely covered
Scrum	Independent, small, self-organized development process. 30-day development lifecycle	The idea is moved from "defined and repeatable" to the "new vision of Scrum for the product development"	Scrum describes in detail how to manage the 30-day cycle but does not focus enough on integration and acceptance testing.
FDD	Five-phase, object oriented, component based development process. Short iterations between a few hours and two weeks.	Simplified methods, design and implementation based on functionalities, object oriented modeling.	FDD is focused mainly on design and implementation but additional supportive processes are also needed.
ASD	Adaptive, collaborative, component based, with a mission, iterative process	Organizations are perceived as adaptive systems. It creates an order in a network of individuals.	It is oriented rather conceptually and to the organization culture than to the software practices.

1.2 Portal Projects. Portal projects are web applications that aggregate information from different sources in a unified way. They offer services like news, stock and products prices, information, e-mail, databases, entertainment, etc. Their development (especially for the public ones) is characterized by the fact that it starts with a certain base of requirements but they begin changing and evolving very quickly. This is due to several factors:

• Unclear vision at the beginning of the project, which starts only with an idea or basic direction of development

• When the first versions of the product are released, the feedback from the real users requires new functionalities or changes to the current ones

• Concurrent products with their specifics, which the portal should take into account.

The constant changes as well as the uncertainty of the success of the future product (will it be accepted by the users) may seriously damage the software

development lifecycle of the product. That is why for these types of starting portals (as well as the companies that develop them) the agile methodologies give an ideal opportunity to answer the arising problems by offering a process which might lead to a successful end of the project in an adaptive way. There are various types of problems that may appear and the different methodologies offer different approaches to handle them. Ketuunen and Laanti [7] have done a split of potential problems when developing embedded software systems and how each methodology addresses them. We will use their ideas as a base for identifying the problems that portal solutions are facing and will make a corresponding analysis. We will list a large set or potential issues without having a concrete portal project in mind. However, it should be taken into account that not all of them may be applicable in an actual project environment and only a subset of them should be considered.

2. Problems in developing portal solutions and approaches offered by the agile methodologies for their solution. The management of modern software projects for development of portal solutions requires very good control over the potential problems, the expected as well as the unexpected situations. Portal projects face many specific problems compared to the other types of software. A powerful tool every project manager should have in order to handle these challenges is the opportunity to choose and eventually later on review the software process. Recently the agile principles have often been pointed out as methodologically suitable for developing projects in this domain. These methodologies put special emphasis on key practices like constant improvement of the project, its initiation, short development cycles, regular feedback and close customer interaction. Their main principles are laid down in the "Agile Manifesto" [8].

We mentioned 4 major methodologies—XP, Scrum, FDD and ASD. The problem faced by the project manager is which of them to choose. We will try to propose a systematic approach in selecting a concrete methodology and practices which would be most suitable under certain conditions. More precisely, we are interested in how the different agile methodologies handle the various problems that appear during the development of portal projects.

We will decompose the software development of a project in three phases—initiation, execution and closure. For each of them we will review the potential difficulties which might be met and how each methodology addresses it. In the classification of the potential problems and their description we will try to give examples of concrete situations and how such issues might appear in a portal project.

2.1. Project Initiation.

• ***Under/Over-estimation in planning.*** Very often the beginning companies who want to break through with their own portal solution have a limited budget and tend to underestimate the price of software development:

— XP is based on continuous planning. Plans are reviewed and adapted according to the last achieved results, the metrics' results shown and the changes in the customer's requirements. It recommends planning at most 2–3 weeks ahead.

— Scrum—the planning is made only before the first iteration based on the requirements already known. At the end of each iteration the results are summarized (Sprint Review) and the next one is planned.

— FDD—overall project planning is not considered. However, FDD still pays attention to the systematic building of the functionalities list and the development plan is based on it. The duration is not estimated as the development of each functionality lasts no more than 2 weeks. The progress of each of them is systematically tracked.

— ASD—the short time boxed development cycles "freeze" the requirements one by one.

• ***Lack of skills.*** Starting a new project with a new customer and developing a solution for which it is still not known if it will be accepted by the end users is a risk for the developing company. For this reason it is not always inclined to commit its best software engineers in the project, at least not before progress becomes visible:

— XP expects the people to be at an expert level. Only some of them may be junior. If the appropriate personnel is not available XP should not be used.

— Scrum depends a lot on the team skills. The staff members should be selected according to their skills and knowledge. The team is responsible for self-organizing the work in the most effective way they see.

— FDD does not address this particular problem. The functionalities are given priorities based on the needs and expectations of the customer and therefore the implementation may still require the technical skill at the phase of defining the project.

— ASD encourages active collaboration in the team and education through iterative product development. Additionally, every team member should develop his/her own skills.

• ***Underestimation of the size, complexity and new technologies in the project.*** In many cases it happens that not everything has been predicted at the beginning of the project or the requirements turn out to be more complex

than initially expected:

— XP—new duration estimation is needed. Pre-planning is a part of XP. The client is always involved in this.

— Scrum—the estimations are often reviewed during the Pregame phase. The project is re-estimated at each Sprint. If it turns out that the expected delivery date is far from the initially envisioned the team should coordinate its activities (Product Backlog) with the customer (Product Owner).

— FDD—a new estimation is needed if it appears that the functionalities are more complex than initially envisioned for finishing the project. Therefore their duration should be short—no more than 2 weeks.

— ASD—portal project are unsure by nature. Everybody should be aware of this since the very beginning. Additional estimations and planning is done after each iteration when more has been learned.

• ***New technology.*** Very often the development of public web applications requires knowledge of new cutting edge technologies in order to cover the increasing vision, functionality and ease of use requirements:

— XP—this often does not fit into XP philosophy for quick planning and simplified design. Knowledge in new technologies is gained "as the project moves on".

— Scrum does not address technology problems specifically. It is possible to have research and prototypes during the planning. The team should self-organize its work so it may approach external experts in order to achieve its goals.

— FDD does not cover this problem.

— ASD—as this is one of the uncertainty points in the project, ASD emphasizes on better knowledge by iterative development.

• ***The project is too big for "at once".*** The project starts with a basic set of requirements, which are too wide scoped and there are many ideas which need to be explored and implemented. Unfortunately there are business deadlines and not everything can be finished on time:

— XP usually deals with small sized projects. The team should not be over 10 developers by definition. Therefore it is difficult to use XP for a bigger scope.

— Scrum—each iteration is fixed at 30 days. This restricts the volume of work to be done. However the number of the iterations is not fixed. Another option is having more than one team working on the project. In this case their work should be coordinated.

— FDD—the project should be split into functionalities that are implemented in steps. All bigger functionalities should be split so as not to exceed 1–2

weeks.

— ASD does not cover this problem.

• ***Extreme project—high speed, many changes.*** Usually it is necessary to complete many tasks and offer the end users a lot of new functionality in short time frames. Very often also changes in the initial requirements and the working process are possible:

— XP is by design created to handle exactly this kind of projects.

— Scrum is oriented towards constant changes but not so much to a high speed. The assumption is that the team is isolated from external pressure. The work expected to be done is agreed in advance and cannot be changed during the Sprint.

— FDD is not intended for extreme cases but a reasonable amount of changes can be accepted. The speed may be increased by developing several functionalities in parallel.

— ASD is by design created to handle exactly such kind of projects.

2.2 Project execution.

• ***Incomplete requirements.*** As is well known requirements very often happen to be not sufficiently clear, specific and detailed in the initial phase of the project. They depend on the operational start of the portal and the feedback from the real users:

— XP expects stable communication with the client. The latter should be on site with the developers including the weekly planning sessions. He is expected to control to what extent the plans correspond to the expectations.

— Scrum expects the client to be on site for the Pregame phase. The Product Owner is responsible for all requirements to be reflected in the Product Backlog. Unclear definition of the requirements in the Backlog might inflict risks to the project.

— FDD—the domain experts work with the functional team in order to explain all problems. Requirements reviews are organized in order to clarify unclear points.

— ASD—ambiguity and lack of initial understanding is considered natural. The goal is to learn more through an iterative development and frequent feedback. It is important to move in the right direction.

• ***Unstable, frequently changing requirements.*** The huge diversity of the potential uses of the portal may result in constant changes in the requirements even for already implemented functionalities:

— XP responds to the changes as something normal. The plan may be changed every week.

— Scrum also accepts the changes as normal. They are reflected in the Product Backlog, which is re-estimated prior each new iteration. However, no changes are allowed during the sprint.

— FDD—a single functionality may be replaced with another, a more complex one. Requirements should be ordered by priority. Up to 10% change is acceptable for a single functionality.

— ASD—development cycles are restricted in time. For the cases with higher uncertainty they should be shorter and unclear items have to be addressed in the beginning.

• ***Constantly changing end date.*** The customer sometimes tends to "postpone" the end date of a release in order to be able to include as much functionality and fixed defects as possible:

— XP—the iterations are fixed in time. If there is a need for a delay new iterations may be added if requested by the client.

— Scrum balances functionalities rather than time. Each Sprint is fixed to 30 days but the number of sprints is not. However, adding a new Sprint is risky.

— FDD— it is expected that functionalities are small and adding new or removing existing ones is easy.

— ASD—the project is fixed in time and the dates of the cycles may not be changed. If it is necessary to change the end date a new planning of the cycles may be negotiated.

• ***Bad/improper architecture.*** In many cases it turns out that the architecture selected in the beginning is inadequate to the constantly expanding scope of functionalities, which may be quite diverse:

— XP does not address specifically this problem. However small changes can be done by using the practice of refactoring, which is a part of XP.

— Scrum—the architecture is defined during the planning phase. The team resolves on its own the uprising technical issues. A mistake in the design may result in a loss of some iteration.

— FDD does not consider this problem. It will be difficult to change the architecture in the middle of the project when most of the functionalities have already been implemented.

— ASD does not cover the architecture design.

• ***Difficulties in integration.*** Every developer in the project may work on a separate part of the code and introduce a problem in the integration with the code of the others:

— XP—all team members should be responsible for the integration, which XP recommends doing on a daily basis.

— Scrum—each Sprint should deliver a working (integrated and tested) package. It is recommended to have daily builds in order to discover early potential communication problems.

— FDD does not explicitly define integration. The developers are responsible for testing their own functionalities. Continuous integration leads to shorter integration steps and therefore the continuous integration becomes easier.

— ASD does not pay special attention to the integration but expects each cycle to end with a working version.

• ***Bad progress tracking.*** It is not always visible to the client what the current progress of the project is:

— XP—this should not be an issue at all as the client is close to the development all the time.

— Scrum—again there is no problem here as the team discusses the progress at the daily meetings and estimates the remaining effort. The Sprint Backlog is updated according to this.

— FDD gives a good visibility of the progress because the delivery of new functionality can be monitored. Progress is tracked based on how much functionality is completed.

— ASD does not give specific practices for tracking the progress of the project as it relies on the collaboration. The only thing that matters is the end result.

• ***Communication gap.*** Whenever there are different people participating in the project it is normal that every one of them has a different level of knowledge, communication skills or is at a different level in the hierarchy compared to his/her colleagues. Differences in the understanding of the requirements between the client and the developers are possible:

— XP is based on frequent and open communication. Gaps in this direction are crucial.

— Scrum is extremely dependent on the team's performance. Therefore it recommends constant communication and exchange of knowledge. For this purpose there are daily meetings within the team. If there is more than one Scrum team special attention should be paid to this.

— FDD domain experts work in close cooperation with the functional teams and this should improve the communication.

— ASD emphasizes on close cooperation. Groups oriented to the client as well as specific inspections are techniques used for learning during the project.

• ***Inadequate documentation.*** Because of the long duration of the portal projects and the variety of participants in them as well as the frequent

changes in the team members and their roles, the documentation becomes an important part of the project. To what extent is it reasonable to focus on it?

— XP prefers working software over documentation.

— Scrum does not prescribe preparation of any documentation. The team is free to document whatever it thinks is necessary without going into details.

— FDD is not oriented towards project documentation. It leaves this do be decided by the project manager according to the specific needs. It puts more attention to the user documentation.

— ASD does not focus on documentation either.

• ***Loss of (key) team members.*** Public portals may be developed and supported for years. It is normal that in such cases some of the people that have started the project or even managed it leave or move to other projects:

— XP—changes in the team reduce the speed of project execution. The departure of a key person is a serious problem.

— Scrum does not consider this a problem as it relies on team members' extreme dedication to their work. This could be a big risk if the team is not able to reorganize after the loss.

— FDD—it may be difficult to replace one class owner with another. Some functional teams may have to be planned again.

— ASD—the review or each cycle pays special attention to the current states of the resources against the goals.

• ***Low motivation.*** New technologies and always pressing end dates require from the people constant adaptation and work under pressure. This may not be to every developer's liking and may result in a decrease of his/her motivation to work:

— XP pays special attention to the motivation of the developers. Avoiding overtime of the 40-hour week is a mandatory measure. Also pair programming may be fun.

— Scrum does not talk about this problem as it expects that the team members are extremely motivated. If this is not the case this may ruin the project as it relies very much on the dedication of the participants.

— FDD recommends using visual tracking of the progress of the functionalities. Presentation of tables and graphics using different colors according to their status may be motivating.

— ASD—building of united groups is one of the cornerstones of ASD. Assuring a suitable environment for this may motivate the people.

2.3 Project Closure.

• ***Problems with acceptance of the system.*** Every project has at the end a phase of User Acceptance Testing. This is the moment at which some of

the issues that have not been properly resolved in the previous steps may pop up as a problem:

— XP develops automated test scenarios which are periodically started. This brings to a better quality and easier integration. These tests are defined and run by the client.

— Scrum is oriented to the client. The results are assessed together with the client (Product Owner) after each Sprint. It may be a problem if unstable versions have been accepted in previous Sprints or if there are multiple teams that encounter difficulties in integrating their work.

— FDD recommends design and code inspections as well as unit testing.

— ASD—technically the quality is controlled during development. It is partially covered by software inspections.

• ***Next version.*** Sooner or later every project or at least some version of it ends. In portal solutions almost always next versions with new functionality are launched. The question is how to use the end of the current phase as a good starting point for the next one:

— XP—the delivery of working software is for sure a good starting point but only this is not enough. XP relies on good knowledge of the project and a change or dismissal of the team may be a serious problem for further development.

— Scrum does not directly discuss this problem. There is a Project Closure phase but the documentation there is not mandatory. This is usually done during the Postgame phase.

— FDD—tables and graphics for tracking the progress of the functionalities give a rough overview of what has been done till the moment. After finishing a certain functionality it may be necessary to prepare user documentation.

— ASD encourages a "strong final" which should be a ground to continue with further versions.

3. Results and comparison. Based on the aforementioned problems faced by the portal solutions we propose a summary of which agile methodology is best at handling each of them. It is contained in Table 2, which lists all the problems discussed, the methodology that best addresses them and the basis on which this decision is made. The different phases have different backgrounds.

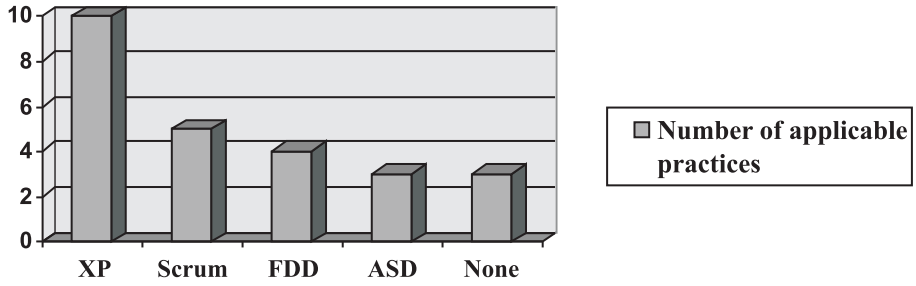
The following Table 3 shows for how many of the problems each methodology is suitable:

It appears that Extreme Programming (XP) provides the biggest number of practices that may be worth using in a portal project in order to handle in an optimal way the potential problems which may appear during the development lifecycle. However, the rest of the methodologies also offer some solutions to this.

Table 2. Suitable methodologies

Problem	AM	Reason
Under/Over-estimation in planning		Based on constant planning
Lack of skills	ASD	Encourages active collaboration and training
Underestimation of the size, complexity and new technologies in the project		Based on constant planning
New technology	—	No methodology covers this problem sufficiently
The project is too big for "at once"	Scrum	May define more than one team working on the project
Extreme project—high speed, lots of changes	XP, ASD	Both methodologies are created to handle precisely this kind of projects
Incomplete requirements	XP	The client is monitoring the project on a weekly basis
Unstable, frequently changing requirements	XP, Scrum	Both methodologies are open to changes and can handle them
Constantly changing end date	FDD	Functionalities are small and it is easy to add new ones
Bad/improper architecture	—	No methodology covers this problem sufficiently
Difficulties in integration	XP, Scrum	Daily builds are recommended
Bad progress tracking	XP, Scrum, FDD	Progress is tracked using the corresponding practices for these methodologies like on-site customer, daily meetings and number of functionalities completed
Communication gap	XP, Scrum	Based on frequent and open communication, on-site customer, pair programming, daily meetings
Inadequate documentation	—	No methodology covers this problem sufficiently
Loss of (key) team members	ASD	The review or each cycle pays special attention of the current states of the resources against the goals
Low motivation	XP, FDD	40-hours week, pair programming, tracking of progress using graphics
Problems with acceptance of the system		Tests are defined and run by the customer
Next version	FDD	Progress graphics are a basis for future development

Table 3. Number of applicable practices



It is important to consider which phase of the project execution each methodology covers.

4. Conclusion. We chose two most widely used (XP, Scrum) and two more theoretically developed (FDD, ASD) agile methodologies for software development and went over the opportunities for their use in the implementation of portal solutions.

We tried to show in a systematic way the concrete problems that may appear with portal projects and to what extent each of the methodologies handles them.

Certainly, each project is individual and all of the potential problems may appear in different situations. It is a responsibility of every project manager to try to identify the risks and threats since the very beginning. He/she should consider that not all of the before mentioned problems may be applicable to his/her project. He/she should select the appropriate subset of them or extend it. After that he/she may use the comparative analysis of the agile methodologies offered by us and its results in order to choose the methodology he/she considers best for implementing the project. Each of them provides opportunities for successful management of the process and a good knowledge of them as well as experience in the development of portal solutions should lead to an optimal choice. However, the project manager is not restricted to selecting a single methodology and can use a combination of practices from all of them in order to implement his/her own agile process.

We consider that the approach proposed by us is generally applicable to other types of software projects as well.

REFERENCES

[1] Artesia—agile methodology definition, http://www.artesiagroup.com/agile-101.html

[2] Beck K. Extreme Programming Explained: Embrace Change. Reading, Mass., Addison-Wesley, 1999.

[3] Schwaber K., M. Beedle. Agile Software Development with Scrum. Upper Saddle River, NJ, Prentice Hall, 2002.

[4] Palmer S. R., J. M. Felsing. A Practical Guide to Feature-Driven Development. Upper Saddle River, NJ, Prentice-Hall, 2002.

[5] Highsmith J. A. Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. New York, NY, Dorset House Publishing, 2000.

[6] Abrahamsson P., O. Salo, J. Ronkainen, J. Warsta. Agile Software Development Methods Review and Analysis, VTT Electronics, Oulu, Finland, 2002.

[7] Kettunen P., L. Maarit. How to Steer an Embedded Software Project: Tactics for Selecting Agile Software Process Models", ICAM 2005 International Conference on Agility, Helsinki, July 27–28, 2005.

[8] Agile Manifesto. http://agilemanifesto.org/

Nikolay Todorov, Avram Eskenazi
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8
1113 Sofia, Bulgaria
e-mail: nikolai.todorov@musala.com
e-mail: eskenazi@math.bas.bg

Received November 19, 2010
Final Accepted January 5, 2011